Global Feature Guided Local Pooling

Takumi Kobayashi
National Institute of Advanced Industrial Science and Technology
1-1-1 Umezono, Tsukuba, Japan
takumi.kobayashi@aist.go.jp

Abstract

In deep convolutional neural networks (CNNs), local pooling operation is a key building block to effectively downsize feature maps for reducing computation cost as well as increasing robustness against input variation. There are several types of pooling operation, such as average/max-pooling, from which one has to be manually selected for building CNNs. The optimal pooling type would be dependent on characteristics of features in CNNs and classification tasks, making it hard to find out the proper pooling module in advance. In this paper, we propose a flexible pooling method which adaptively tunes the pooling functionality based on input features without manually fixing it beforehand. In the proposed method, the parameterized pooling form is derived from a probabilistic perspective to flexibly represent various types of pooling and then the parameters are estimated by means of global statistics in the input feature map. Thus, the proposed local pooling guided by global features effectively works in the CNNs trained in an end-to-end manner. The experimental results on image classification tasks demonstrate the effectiveness of the proposed pooling method in various deep CNNs.

1. Introduction

Deep convolutional neural networks (CNNs) are successful models for high-performance image recognition [18, 32, 12]. While some techniques such as Batch-Normalization [16] and DropOut [33] are fundamental to properly train the deep models, from the architectural viewpoint, the CNNs mainly comprise three basic operations of convolution, activation and spatial pooling. The activation functions, especially rectified-linear unit (ReLU) [26] and its variants [24, 11], are applied to non-linearly transform the neuron responses. The convolution is a crucial operation to extract effective features from an input image by the learned filters [44, 4, 41]. The *local* convolution is expanded to fully-connected (FC) one [18] which performs *globally*.

In a similar way to the convolution, there are two types of spatial pooling in terms of receptive field, *local* or *global* ones. The *global* pooling effectively substitutes for the FC in some CNNs [22, 12, 34] through spatially compressing the (last) feature map of space-channel tensor into the feature vector of channel dimensionality which is finally fed into classification layers. Practically speaking, the average-pooling is mainly applied to globally aggregate features, though some pooling forms are also investigated in [17].

In contrast to the global pooling, *local* pooling operation is a key building block commonly employed in most CNNs to efficiently reduce spatial resolution with increasing robustness against variations in input images, such as translation. The local pooling also stems from the biological insight [15]. According to the biologically-inspired model [31], various types of deep CNNs [18, 32, 34] employ local max-pooling for downsizing the feature maps, while average-pooling is also applied to CNNs [20]. On the other hand, some models [12, 41] achieve the same effect of downsizing by means of *strided* convolution which is also regarded as a pooling following the convolution of 1-striding [46]. Therefore, in contrast to convolution operation, there are several ways, such as avg/max, to implement the pooling operation and it is hard to manually choose the optimal pooling type; it is determined based on the empirical performance, requiring huge amounts of effort in a trial and error approach. Thus, it motivates us to optimize the type of pooling function in an end-to-end training, as is done for convolution filters or leaky-ReLU parameters [11].

Toward trainable pooling operation, it is necessary to address two issues regarding (1) how to formulate various types of local pooling function and (2) what kind of data to use for determining the pooling type. Diverse pooling functionality has to be represented by a simple and unified form, and the pooling operation in the form should be adaptively tuned based on the input features even on a test phase, since the optimal pooling type is related to the characteristics of input features [1, 38]. Thus, the trainable pooling demands such a dynamic and flexible formulation. In addition, the optimal pooling functionality would be determined based on the *global* characteristics of input features beyond *local*

receptive field [1, 38]. The local and global pooling have been discussed separately as above, and there is no fusion between them; the local pooling function so far deals with only local features in the receptive field [30, 21].

In this work, we propose a novel trainable *local* pooling function guided by the *global* features beyond the local ones. We first formulate the pooling function based on the maximum entropy principle [25] to flexibly represent various types of pooling functions with trainable parameters. The type of pooling functionality is effectively controlled by the parameters. Then, we leverage global feature statistics to estimate the parameters of the flexible pooling function. In the proposed pooling method, the parameters are not directly trained in a static form but dynamically determined by means of the global features, in contrast to the other parametric (trainable) pooling methods [30, 21]. Our main contributions are three folds: (1) we theoretically derive the parameterized pooling function which flexibly describes various types of pooling, (2) incorporate the global features into the pooling operation through adaptively estimating the pooling parameters, and (3) perform thorough experiments on large-scale datasets to present the effective pooling form by analyzing the method from various aspects, while showing the favorable performance in comparison with the other methods.

2. Related Works

The spatial pooling originates from the biological work about complex cells in the mammalian visual cortex [15]. Then, the importance of max-pooling has been discussed in some neuroscientific works through analyzing/mimicking the primary visual cortex area V1 [28, 29, 31].

The pooling is also applied for rather practical purpose to aggregate features locally to build the local image descriptors, such as SIFT [23] and HOG [5], via average-pooling. In the framework of bag-of-words [3], the spatial pooling is applied globally to aggregate word codes assigned to the local descriptors toward effective image representation. While average-pooling is widely applied to count words, max-pooling also works well in conjunction with sparse coding techniques [1, 42].

In the literature of neural networks, a pooling technique is applied either to summarize neuron activations across *channel* in a multi-layered perceptron (MLP) or to downsize *spatial* resolution in a convolutional neural network (CNN). In the *channel* pooling, L_p-norm is discussed from the viewpoint of signal recovery [2] and is extended to the trainable one through learning the parameter p [9] to smoothly transit from average ($p = 1$) to max ($p = \infty$) operation. The trainable L_p pooling is also related to MAXOUT [7] which performs inter-channel max-pooling.

The local *spatial* pooling is widely applied to deep CNNs for gradually downsizing the feature maps with increasing computation efficiency and robustness. While the average pooling is employed in some CNNs [20], the max pooling becomes a popular choice for deep CNNs [18, 32, 34] according to the biologically-inspired model of HMAX [31]. Nonetheless, in building CNNs, it is necessary to determine the type of pooling, average or max, and the optimal pooling type would be dependent on the recognition tasks and/or characteristics of the input features [1]. There are works [43, 21, 30] to address the problem by bridging the gap between these two pooling operations, with similar motivation to ours. The average and max pooling functions can be simply integrated in a convex combination with one weighting parameter which is randomly selected in [43] or trained based on the local features by [21]. To mitigate the artifacts caused by downsizing, the image processing techniques are also applied to the pooling [40, 30]. Wavelet pooling [40] employs the wavelet to compress a feature map accurately with less artifacts. The detailed preserving pooling (DPP) [30] is also proposed based on the image downscaling technique [39] to formulate a parametric pooling function as an intermediate operation between average and max. The proposed method is close to those methods [21, 30] but is clearly different in the following two points; the parametric pooling function is derived in a theoretical way based on maximum entropy principle [25] to flexibly represent various types of pooling including average/max-pooling (Sec. 3), and the parameters are dynamically estimated from the global features beyond the local receptive field of the pooling (Sec. 4).

From the probabilistic viewpoint, stochastic approaches are applied to the local spatial pooling function [45, 46, 43] for introducing randomness into the CNNs as in DropOut [33]. In contrast to the deterministic pooling mentioned above, the stochastic methods randomly pick up a neuron activation in the receptive field throughout an end-to-end training. We also discuss the connection between the proposed method and the stochastic pooling in Sec. 3.3.

3. Parameterized Pooling Function

The local spatial pooling is generally formulated as follows. Given an input feature map $\boldsymbol{X} \in \mathbb{R}^{H \times W \times C}$, the c-th channel output Y_p^c at the pixel p is computed by applying the spatial pooling to the c-th channel feature map as

$$Y_p^c = \sum_{q \in \mathcal{R}_p} \mathtt{W}_p^c(q, \boldsymbol{X})\, X_q^c, \tag{1}$$

$$s.t.\ \mathtt{W}_p^c(q, \boldsymbol{X}) \geq 0,\ \forall q \in \mathcal{R}_p,\ \sum_{q \in \mathcal{R}_p} \mathtt{W}_p^c(q, \boldsymbol{X}) = 1,\ \forall p, c,$$

where $\mathcal{R}_p$ is a set of pixel positions in the receptive field (neighborhood) of the pixel p, and $\mathtt{W}_p^c$ is a weighting function to represent the type of pooling. For example, *average-pooling* [20] is realized by $\mathtt{W}_p^c(q, \boldsymbol{X}) = \frac{1}{|\mathcal{R}_p|}$ in disre-

gard of both the position q and the features $\boldsymbol{X}$, and *max*-pooling [32] is represented by $\mathtt{W}^c_p(q^*, \boldsymbol{X}) = 1, \mathtt{W}^c_p(q \neq q^*, \boldsymbol{X}) = 0$ where $q^* = \arg\max_{q \in \mathcal{R}_p} X^c_q$, while *skipping* [12] is simply given by $\mathtt{W}^c_p(q = p, \boldsymbol{X}) = 1, \mathtt{W}^c_p(q \neq p, \boldsymbol{X}) = 0$. Thus, designing pooling operation results in how to formulate the weighting function $\mathtt{W}^c_p$.

3.1. Maximum entropy principle

In this work, based on the constraints in (1), we regard the weighing function $\mathtt{W}^c_p$ as a probability density function, and from the probabilistic perspective, formulate the following optimization problem for $\mathtt{W}^c_p$:

$$\max_{\mathtt{W}} \sum_{q \in \mathcal{R}_p} -\mathtt{W}(q)\log[\mathtt{W}(q)] + \tilde{\lambda}\mathtt{W}(q)X_q - \eta \mathtt{W}(q)\log\left[\frac{\mathtt{W}(q)}{\tilde{\rho}_q}\right], \tag{2}$$

$$s.t.\ \mathtt{W}(q) \geq 0,\ \sum_{q \in \mathcal{R}_p} \mathtt{W}(q) = 1, \tag{3}$$

where we omit in $\mathtt{W}$ the notations of p, c and $\boldsymbol{X}$ for simplicity and introduce regularization parameters $\tilde{\lambda} \in \mathbb{R}$ and $\eta > 0$ as well as position prior distribution $\{\tilde{\rho}_q\}_{q \in \mathcal{R}_p}$. The first term of (2) is derived from the maximum entropy principle [25] which is a natural assumption to determine a probabilistic model via an optimization problem. In the second and the third terms, two regularizations regarding the output Y^c_p (1) and the position are introduced from the viewpoint of pooling functionality. The second term intends to make the output $Y = \mathtt{W}(q)X_q$ more distinctive via maximization ($\tilde{\lambda} > 0$) or minimization ($\tilde{\lambda} < 0$). And, the weighing $\mathtt{W}$ should be close to the predefined position priors $\{\tilde{\rho}_q\}$ via minimizing Kullback–Leibler divergence in the third term.

(2) can be solved by introducing the Lagrange multipliers $\alpha_q \geq 0$ and $\beta \in \mathbb{R}$ for the non-negativity and unit-sum constraints in (3) to provide the derivative w.r.t. $\mathtt{W}(q)$ as

$$-(1+\eta)(1+\log[\mathtt{W}(q)]) + \tilde{\lambda}X_q + \eta\log[\tilde{\rho}_q] + \alpha_q + \beta = 0, \tag{4}$$

which leads to the following form of $\mathtt{W}$,

$$\mathtt{W}(q) = \exp\left\{\frac{1}{1+\eta}(\tilde{\lambda}X_q + \eta\log[\tilde{\rho}_q] + \alpha_q + \beta) - 1\right\} \tag{5}$$

$$= \frac{\exp(\lambda X_q + \rho_q)}{\sum_{q' \in \mathcal{R}_p} \exp(\lambda X_{q'} + \rho_{q'})}, \tag{6}$$

where $\alpha_q = 0$ due to the positivity of (5) and the KKT condition, and β is defined so as to satisfy the unit-sum constraint. We also reparameterize $\lambda = \frac{\tilde{\lambda}}{1+\eta} \in \mathbb{R}$ and $\rho_q = \frac{\eta}{1+\eta}\log[\tilde{\rho}_q] + \epsilon$ with some constant ϵ to let $\rho_q \in \mathbb{R}$ without loss of generality. The weighting function $\mathtt{W}$ is decomposed into $\mathtt{W}(q) \propto \exp(\lambda X_q)\exp(\rho_q)$ which comprises two kinds of weights regarding feature X and spatial position q, as in the bilateral filter [35]. Finally, we obtain the parameterized pooling function by

$$Y^c_p = \frac{\sum_{q \in \mathcal{R}_p} X^c_q \exp(\lambda^c X^c_q + \rho^c_{q-p})}{\sum_{q \in \mathcal{R}_p} \exp(\lambda^c X^c_q + \rho^c_{q-p})}, \tag{7}$$

where we make the parameters independent of the position p and dependent only on the channel c. Namely, the pooling function contains the parameters of $\boldsymbol{\lambda} = \{\lambda^c\}^C_{c=1} \in \mathbb{R}^C$ and $\boldsymbol{\rho} = \{\rho^c_{q-p}\}^C_{c=1, q \in \mathcal{R}_p} \in \mathbb{R}^{C|\mathcal{R}_p|}$ where 'q–p' means the relative position from p in the receptive field $\mathcal{R}_p$. Thereby, the parameters are shared across any position p at that layer.

3.2. Derivative

The derivatives of the pooling function (7) is given by

$$\frac{\partial Y^c_p}{\partial X^c_q} = \frac{\exp(\lambda^c X^c_q + \rho^c_{q-p})}{\sum_{q' \in \mathcal{R}_p} \exp(\lambda^c X^c_{q'} + \rho^c_{q'-p})}\left\{1 + \lambda(X^c_q - Y^c_p)\right\}, \tag{8}$$

$$\frac{\partial Y^c_p}{\partial \lambda} = \frac{\sum_{q \in \mathcal{R}_p}(X^c_q - Y^c_p)^2 \exp(\lambda X^c_q + \rho^c_{q-p})}{\sum_{q' \in \mathcal{R}_p} \exp(\lambda^c X^c_{q'} + \rho^c_{q'-p})}, \tag{9}$$

$$\frac{\partial Y^c_p}{\partial \rho_q} = \frac{(X^c_q - Y^c_p)\exp(\lambda X^c_q + \rho^c_{q-p})}{\sum_{q' \in \mathcal{R}_p} \exp(\lambda^c X^c_{q'} + \rho^c_{q'-p})}, \tag{10}$$

where we consider the derivatives w.r.t. the input feature X^c_q as well as the two parameters λ^c and ρ^c_{q-p}, all of which can be trained through back-propagation in an end-to-end manner. The parameter λ^c is updated by (9) based on the variance of the features, adapting to the scale of the features. On the other hand, the derivative (8) is summed up to 1 on the receptive field $q \in \mathcal{R}_p$, as is the case with the standard pooling such as avg/max-pooling. The update is distributed over the receptive field according to the contribution measured by $\frac{\exp(\lambda^c X^c_q + \rho^c_{q-p})}{\sum_{q' \in \mathcal{R}_p} \exp(\lambda^c X^c_{q'} + \rho^c_{q'-p})}$, suppressing the features that are significantly smaller than the average Y^c_p, as in *Lateral inhibition* [8], to let the network extract diverse features.

3.3. Discussion

Flexibility. The pooling function (7) parameterized by $\boldsymbol{\lambda}$ and $\boldsymbol{\rho}$ flexibly describes various types of pooling functions; *average* and *max* pooling are produced by $\{\lambda = 0, \rho_{q-p} = 0\}$ and $\{\lambda \to \infty, \rho_{q-p} = 0\}$, respectively, while $\{\lambda = 0, \rho_{q-p} = \delta_{q-p}\}$ leads to *skipping*. In addition, we can also realize *min*-pooling by $\{\lambda \to -\infty, \rho_{q-p} = 0\}$. The position prior, which endows the pooling with local position sensitivity, can also be related to a wavelet filter in the Wavelet pooling [40]. Such a flexibility of the parametric pooling (7) is a key property for adaptively controlling the pooling type via the global features (Sec. 4).

Softmax. Without position priors, *i.e.*, $\boldsymbol{\rho} = \mathbf{0}$, the pooling (7) corresponds to the α-softmax [19], which is mentioned in the literature of neuroscience [29, 27] and in the

framework of bag-of-words [1]. And, the parameter λ is connected to the scaling factor in L_2-normalized softmax [37] and to the temperature of softmax in the other literatures [13, 10]; λ is a reciprocal of the temperature, $\lambda = \frac{1}{T}$. In those methods, the temperature is tuned by hand to properly transfer the network structure in [13] and is discriminatively learned based on labeled data for fine-tuning the confidence of the classifier in [10] or for improving the classification performance in the l_2-normalized softmax [37]. In contrast, we naturally derive the parameter λ from the optimization problem (2) based on the maximum entropy principle. Through controlling the two terms of the entropy and the significance of the output in (2), the parameter λ plays a role in smoothly switching the pooling functionality from average to extreme ones (min/max). In addition, the proposed method is distinctive in that the pooling parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\rho}$ in (7) are adaptively determined by means of global features as described in Sec. 4.

Probabilistic viewpoint. As described in Sec. 3.1, the pooling (7) outputs the probabilistic mean of the local features according to the probability density function $\mathtt{W}$ in (6). On the other hand, the stochastic pooling [45] picks up the feature from the receptive field $\{X^c_q\}_{q\in\mathcal{R}_p}$ according to the probability proportional to the non-negative feature values, $p_q \propto X^c_q$, which is related to (6). For small λ^c and $\boldsymbol{\rho} = \mathbf{0}$, we can approximate $\exp(\lambda^c X^c_q) \approx 1 + \lambda^c X^c_q$, and thereby the pooling weights (6) result in the form of $\mathtt{W}(q) \approx \frac{1+\lambda^c X^c_q}{|\mathcal{R}_p|+\lambda^c \sum_{q'\in\mathcal{R}_p} X^c_{q'}}$ corresponding to the biased probability of [45]. And, we can say that the form (6) is more flexible since it is applicable to any features while the stochastic pooling [45] accepts only non-negative features produced such as by ReLU.

4. Global Feature Guidance

We leverage the *global* features to estimate the *parameters* of $\boldsymbol{\lambda}$ and $\boldsymbol{\rho}$ which control the type of pooling function in (7) operating *locally* on the receptive field $\{X^c_q\}_{q\in\mathcal{R}_p}$. For that purpose, the pooling parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\rho}$ are regarded as *variables* rather than static *parameters* to be optimized in the training. We let the variables be dependent on the input features $\boldsymbol{X} \in \mathbb{R}^{H\times W\times C}$ and thus described by the mapping $\boldsymbol{\lambda} = \mathtt{f}(\boldsymbol{X})$ and $\boldsymbol{\rho} = \mathtt{g}(\boldsymbol{X})$. Following the methodology in squeeze-and-excitation (SE) [14], the mapping functions $\mathtt{f}$ and $\mathtt{g}$ are designed by means of the multilayer perceptron (MLP) applied to the global feature statistics (Fig. 1);

$$\boldsymbol{\lambda} = \mathtt{f}(\boldsymbol{X}; \boldsymbol{U}, \boldsymbol{V}_\lambda) \tag{11}$$

$$= \mathtt{s}\big(\boldsymbol{V}_\lambda\, \mathtt{ReLU}(\boldsymbol{U}\,[\mathtt{t}(\{X^1_p\}_{\forall p}), \cdots, \mathtt{t}(\{X^C_p\}_{\forall p})]^\top)\big), \tag{12}$$

where we consider one hidden layer of D neurons with ReLU and the function $\mathtt{t}$ computes k types of statistics per channel over the inputs $\{X^c_p\}^{(H,W)}_{p=(1,1)}$ and $\mathtt{s}$ is an element-wise non-linear activation function; for example, we can set

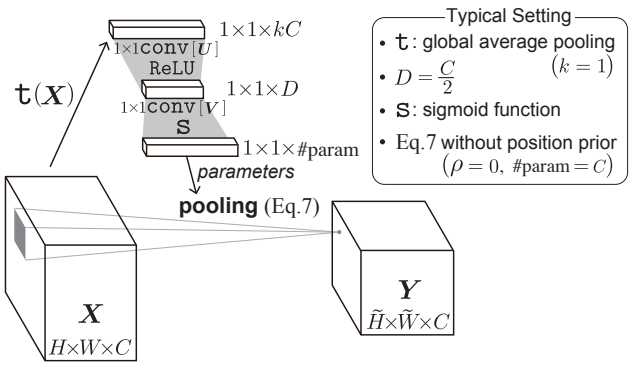


Figure 1. Global Feature Guided Pooling (GFGP).

$\mathtt{t}$ to a global averaging function ($k = 1$) and $\mathtt{s}$ to a sigmoid function as in [14]. The similar MLP is applied for $\boldsymbol{\rho}$ by $\mathtt{g}(\boldsymbol{X}; \boldsymbol{U}, \boldsymbol{V}_\rho)$ which shares $\boldsymbol{U}$ with $\mathtt{f}$, though the activation $\mathtt{s}$ may be different. Thus, in the proposed pooling, called *global feature guided pooling* (**GFGP**), the MLP weights $\boldsymbol{U} \in \mathbb{R}^{D\times kC}$, $\boldsymbol{V}_\lambda \in \mathbb{R}^{C\times D}$ and $\boldsymbol{V}_\rho \in \mathbb{R}^{|\mathcal{R}_p|C\times D}$ are the targets to be optimized in an end-to-end training.

In the proposed method, the key point is to deal with the pooling parameters as **variables** to be mapped from the global features by (11). Thus, from that viewpoint, it contrasts with the other methods, as follows.

Constant. The pooling with pre-fixed constant λ produces such as avg-pooling ($\lambda = 0$), max-pooling ($\lambda = \infty$) and the intermediate one between them [1] ($0 < \lambda < \infty$). In contrast to those pre-fixed pooling, the proposed method determines the type of pooling adaptively based on data without manually tuning it, as in the parameterized pooling [30, 21].

Parameter. In [30], the pooling parameters are directly trained in an end-to-end manner. Thus, the trained pooling types could be varied across both channels and layers adaptively unlike the above-mentioned constant pooling. However, the pooling types, *i.e.*, pooling parameters, are fixed once trained, for a test phase. On the other hand, the proposed pooling is dynamically dependent on the input features $\boldsymbol{X}$ via the mapping (11); that is, the same pooling layer works differently according to the input features, which exhibits distinctiveness compared to the previous parameterized pooling [30].

Locality. While the gated pooling [21] also estimates the gating parameter via the features yet *locally*, the functionality of the proposed pooling is determined based on the *global* characteristics of input features and works in the *local* receptive field. Thus, we can say that the proposed method incorporates both *local* and *global* information into the pooling. The pooling is generally applied to downsize the spatial resolution of an input feature map, inevitably loosing (spatial) information, and the global information in the proposed pooling would compensate it for improving performance.

5. Experimental Results

We evaluate the performance of the proposed method by embedding it into the deep CNNs trained on large-scale datasets for image classification [6, 48]. The classification performance is measured by single-crop top-1 and top-5 error rates (%) on the validation set of dataset.

5.1. Ablation study

We analyze the proposed pooling method (7, 11) on ImageNet classification [6]. It is applied to the deep CNN of VGG-13 [32] which is slightly modified from the original model [32] by introducing Batch-Normalization [16] and reducing the number of channels in the FC layers from 4096 to 2048. The CNN contains *five* local max-pooling layers with the pool size of 2×2 and $(2,2)$-striding to downsize the feature map resolution by a factor of 2. We replace all the local pooling layers by the proposed method and train the CNN by following the learning protocol provided in MatConvNet [36]. Unless otherwise noted, the proposed pooling method is implemented by the default setting (Fig. 1) which applies the sigmoid $\mathtt{s}$ and the global average pooling $\mathtt{t}(\{X_p^c\}) = \frac{1}{HW}\sum_{p=(1,1)}^{(H,W)} X_p^c$ to the MLP of $D = \frac{C}{2}$ without position priors $\boldsymbol{\rho} = \mathbf{0}$; these are discussed in Sec. 5.1.2.

5.1.1 Global feature guiding model

We clarify how the proposed scheme of *global feature guidance* (Sec. 4) contributes to improving the pooling. As mentioned in Sec. 4, we can consider the three types of the pooling function regarding λ in (7) ; we apply *constant* $\lambda = 1$ as the simplest way, directly train the *parametric* λ as in the trainable pooling [30], and estimate the *variable* λ from the global features by the proposed GFGP (11). As shown in Table 1a, the methods that learn λ as *parameter* and *variable* work well to improve performance, and particularly the GFGP leveraging the global information to *variable* λ via (11) outperforms the others by a large margin. Then, the importance of the *global* information in GFGP is verified through the comparison with the method that applies (11) *locally*; that is, in (11), $\mathtt{t}$ is set to a *local* average pooling on the 2×2 receptive field to produce the variable λ_p^c for each output Y_p^c. Note that this *local* method also employs the same MLP model composed of the parameters $\boldsymbol{U}$ and $\boldsymbol{V}_\lambda$ of the same size as in the GFGP (11). The performance comparison in Table 1b demonstrates that the global method (GFGP) is superior to the local one, validating the effectiveness of the global information in the pooling method.

While the position prior was turned off ($\boldsymbol{\rho} = \mathbf{0}$) in the above analysis, we here apply the full GFGP to estimate two variables of $\boldsymbol{\lambda}$ and the position priors $\boldsymbol{\rho}$; in this case of 2×2 pool size, the position priors are described by *four*-dimensional variable per channel resulting in $\boldsymbol{\rho} \in \mathbb{R}^{4C}$. Table 1c shows that the full method is comparable to the simple method of $\boldsymbol{\rho} = \mathbf{0}$. The sensibility to the local position via the priors $\boldsymbol{\rho}$ might prevent the pooling from increasing robustness against translation. To remedy it, the position prior ρ can also be estimated locally as in Table 1b while λ is globally estimated. It slightly improves performance as 28.53% (top-1) and 9.73% (top-5), though significantly increasing computation cost for the local estimation $\{\rho_{q-p}\}_{p,q}$. Thus, in this work, we apply the simple GFGP without priors $\boldsymbol{\rho}$ in (7) which contains only C-dimensional variable $\boldsymbol{\lambda}$ to be estimated by (11).

Table 1. Performance analysis for global feature guidance.

(a) Pooling type w.r.t. λ in (7) under $\rho = 0$

λ	Constant	Parameter	Variable (GFGP)	original max-pool
Top-1	29.82	29.56	28.57	30.02
Top-5	10.40	10.26	9.65	10.27

(b) Receptive field in (11)

$\mathtt{t}$	global avg	local avg
Top-1	28.57	29.02
Top-5	9.65	9.92

(c) Position prior

Variable	λ	$\lambda, \{\rho_q\}_q$
Top-1	28.57	28.62
Top-5	9.65	9.57

5.1.2 MLP model in GFGP

Next, we analyze the MLP model (12) to map the global features into the pooling parameters $\{\lambda^c\}_{c=1}^C$.

Global statistics $\mathtt{t}$. There are three standard models of global *average (avg)*, *standard deviation (std)* and *max* for the statistics $\mathtt{t}$, which are compared in Table 2a; note that the statistics of *avg+std* produce doubled feature dimensionality ($k = 2$ in Fig. 1) compared to the others ($k = 1$). The simple global *avg* exhibits favorable performance, while *max* would be less suitable to these dense feature maps [1] and be vulnerable to outliers in the less discriminative features at the shallower layers.

MLP architecture. Table 2b compares the MLPs of various numbers of hidden nodes D as well as the single layered perceptron (SLP) of $\mathtt{s}(\boldsymbol{U}\mathtt{t}(\boldsymbol{X}))$. Under the same number of total parameters, C^2, the MLP-$\frac{1}{2}$ outperforms the SLP due to the non-linearity of the MLP mapping. Though MLP-1 doubling the number of hidden nodes D slightly improves the performance, the MLP-$\frac{1}{2}$ is favorable based on the trade-off between performance and computation cost.

Activation function $\mathtt{s}$. The range of λ is restricted via the activation function $\mathtt{s}$, for which we can consider four types of functions; sigmoid $\frac{1}{1+\exp(-x)} \in [0,1]$, softplus $\log(1+\exp(x)) \in [0,+\infty]$, hyperbolic tangent (tanh) $\frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)} \in [-1,1]$, and identity $x \in [-\infty,+\infty]$. Note that the latter two functions might push the pooling toward *min*-pooling via $\lambda < 0$. The scale of λ is bounded by tanh and sigmoid functions, letting the scale factor in (7) rely on the features $\boldsymbol{X}$ which would be properly trained in an end-to-end manner; if the features are normalized, the scale factor should be embedded into λ as in [37]. We

Table 2. Performance comparison across various settings in the MLP model (12).

(a) Global statistics $\mathtt{t}$

Statistics	avg	avg+std	max
Top-1	28.57	28.56	28.93
Top-5	9.65	9.70	10.01

(b) Mapping architecture

Architecture	SLP	MLP-$\frac{1}{4}$	MLP-$\frac{1}{2}$	MLP-1
size of $\boldsymbol{U}$	$C \times C$	$\frac{C}{4} \times C$	$\frac{C}{2} \times C$	$C \times C$
size of $\boldsymbol{V}$	-	$C \times \frac{C}{4}$	$C \times \frac{C}{2}$	$C \times C$
Top-1	28.72	28.80	28.57	28.36
Top-5	9.71	9.83	9.65	9.51

(c) Activation function $\mathtt{s}$

$\mathtt{s}$	sigmoid	softplus	tanh	identity
Range of λ	$[0, 1]$	$[0, +\infty]$	$[-1, 1]$	$[-\infty, +\infty]$
Top-1	28.57	28.63	28.96	28.70
Top-5	9.65	9.75	9.75	9.80

can see in Table 2c that the sigmoid and softplus functions work well, producing non-negative λ to push the pooling (7) toward avg/max-pooling. This result implies that the min-pooling suppressing the feature channel degrades performance, while avg/max-pooling excites the feature channel to favorably improve the performance. The best performance is produced by the sigmoid which properly limits the range of λ as well as excludes the min-pooling.

5.1.3 Effectiveness by increased number of parameter

Our pooling layer is efficiently computed due to the global average pooling followed by the MLP as in SE [14], computation cost of which might be further improved such as by grouped convolution [41] and channel shuffling [47].

As to the network size, the proposed pooling method introduces only C^2 additional parameters per pooling layer, as shown in Table 2b. From the viewpoint of the increased number of network parameters, we show the effectiveness of the proposed method in comparison with the other types of layers that adds the same number of parameters; NiN [22] based on 1×1 conv, ResNiN which adds an identity path to the NiN module as in ResNet [12], and squeeze-and-excitation (SE) [14], whose detailed structures are shown in the supplementary material. For fair comparison, these methods are implemented by using the same MLP-$\frac{1}{2}$ as ours (Table 2b) of C^2 parameters and are embedded so as to work on the feature map fed into the (original) *max* pooling layer; note that those comparison modules do not provide the functionality of downsizing feature map. The performance results are shown in Table 3, demonstrating that the proposed pooling method benefits from the additional parameters most effectively. And, it should be noted that for further improving performance, the proposed pooling method could work in conjunction with these modules that enhance the discriminativity of the feature map without pooling functionality.

Table 3. Performance comparison under the same number of additional parameters, C^2 per pooling layer.

Model	NiN [22]	ResNiN [22]	SE [14]	Ours
Top-1	30.61	29.12	29.24	28.57
Top-5	10.80	10.04	10.03	9.65

Table 4. Performance analysis regarding depths where the proposed pooling substitutes for the original max-pooling. '✓' indicates the replacement by the proposed one, while '-' means the original max-pooling.

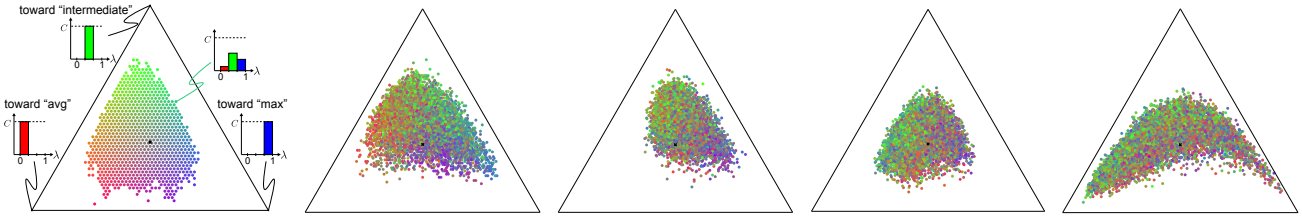
	Pooling layer (Depth)					Error rate (%)	
	1st	2nd	3rd	4th	5th	Top-1	Top-5
(i)	✓	-	-	-	-	30.01	10.47
(ii)	✓	✓	-	-	-	29.68	10.38
(iii)	✓	✓	✓	-	-	29.43	10.24
(iv)	✓	✓	✓	✓	-	29.08	9.89
(v)	✓	✓	✓	✓	✓	28.57	9.65
(vi)	-	✓	✓	✓	✓	28.69	9.61
(vii)	-	-	✓	✓	✓	29.04	9.83
(viii)	-	-	-	✓	✓	28.92	9.90
(ix)	-	-	-	-	✓	29.71	10.17

5.1.4 Depth

We investigate the effect of our pooling method in terms of the depth where it is embedded. In this experiment, the method replaces some of the max-pooling layers in VGG-13, while in the other experiments the CNN is fully equipped with the proposed method at all the pooling layers. The performance results in Table 4 show that our method at the deeper layers contributes to performance improvement more effectively; removing the proposed pooling at the shallower layers (Table 4vi∼viii) degrades performance only slightly. The deeper layers produce the more discriminative features on which the flexible pooling works well. Thus, it would be effective to apply the proposed pooling method only at the deeper layers, for suppressing the increase of parameters in CNNs.

5.2. Analysis of variable $\boldsymbol{\lambda}$

We then analyze the contents of $\boldsymbol{\lambda}$ produced by (11) in the proposed GFGP. At each local pooling layer, the produced $\{\lambda^c\}_{c=1}^C$ are distributed over $[0, 1]$ due to the sigmoid activation $\mathtt{s}$ and we quantize the distribution into three-dimensional histogram of three bins as shown in Fig. 3. Since the sum of the histogram counts is constantly C, the three-dimensional histograms are lying on 2-simplex as shown in Fig. 2; we randomly picked up 10,000 samples from the ImageNet dataset and feed-forward them through the trained VGG-13 to obtain 10,000 histogram vectors at each pooling layer. At the first layer, one can see various



1st layer ($C = 64$) 2nd layer ($C = 128$) 3rd layer ($C = 256$) 4th layer ($C = 512$) 5th layer ($C = 512$)

Figure 2. Visualization of the distribution of $\{\lambda^c\}_{c=1}^C$ produced by (11) where C is the number of channel at each pooling layer. The distribution of C samples $\{\lambda^c\}_{c=1}^C$ is quantized into three bins to form three-dimensional histogram (Fig. 3) which lies on 2-*simplex*. Each point indicates each distribution (histogram of three bins) and is colored by assigning to RGB-channels the frequencies on the three histogram bins at the first layer. The center point ('$\times$') means a uniform distribution. This figure is best viewed in color.

types of the distribution of λ which are spread diversely around the center[1]. It indicates that the optimal pooling types vary from sample to sample. On the other hand, at the deeper layers, the distribution of λ is somehow biased, and in particular, the last fifth layer exhibits high bias toward the pooling type of either average or max. The last pooling layer receives discriminative features which might require the distinctive pooling type such as average/max-pooling. It would lead to the effectiveness of the proposed pooling at the deeper layers as discussed in Sec. 5.1.4.

Then, to measure the dependency of λ on the object categories, we show in Table 5a the Fisher discriminant score by applying the Fisher discriminant analysis to the three-dimensional histogram vectors which are provided with the class labels. All the layers exhibit low discriminant scores, indicating that the produced λ are less dependent on object categories. Actually, in Fig. 3 which shows examples of the distribution of λ, we can see clear difference between two samples belonging even to the identical ImageNet category.

We also analyze the relationship between layers in terms of λ. Each sample image produces the distributions of λ at respective layers, and the correlation coefficients between layers are computed by canonical-correlation analysis over the three-dimensional histogram vectors. As shown in Table 5b, although the adjacent two layers exhibit relatively high correlations, they are generally low, showing less correlation among the layers.

Thus, we can conclude that it is important to produce the variable $\boldsymbol{\lambda}$ at each layer for each sample without sharing it across layers nor considering class categories.

5.3. Comparison to the other methods

Finally, the proposed method is compared with the other pooling methods on several deep CNNs. We first consider the CNNs of VGG-16 [32] which contains five local max-pooling layers and VGG-16-mod [17] of four local max-pooling layers and one global average-pooling; all the local *max*-pooling layers are implemented with 2×2 pool size and $(2, 2)$-striding. We replace those local pooling with

[1]The center point corresponds to the uniform distribution of $\{\lambda^c\}_{c=1}^C$.

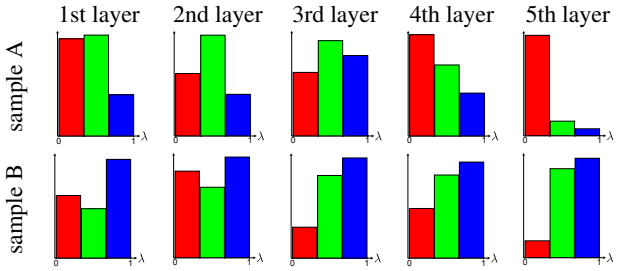


Figure 3. Examples of the distributions of $\{\lambda^c\}_{c=1}^C$. The distribution is quantized into a histogram of three bins over $[0, 1]$. These two histograms are produced by two image samples belonging to the identical ImageNet category.

Table 5. Statistics for the distribution of λ. (a) The higher score means the higher class-dependent distribution of λ. (b) The higher correlation indicates the closer connection regarding the distribution of λ between two layers.

(a) Fisher discriminant score

1st layer	2nd layer	3rd layer	4th layer	5th layer
0.1395	0.2021	0.1854	0.2276	0.1610

(b) Correlation coefficient

	1st layer	2nd layer	3rd layer	4th layer	5th layer
1st	-	0.4400	0.2451	0.1739	0.1445
2nd	-	-	0.3172	0.2165	0.1771
3rd	-	-	-	0.3454	0.2581
4th	-	-	-	-	0.3795

the other types of pooling for comparison: *skip* pooling implemented by the convolution with $(2, 2)$-striding, *average*-pooling, two types of *stochastic* pooling methods [45, 46], three trainable pooling methods of *DPP* [30], *Gated* pooling [21] and our pooling (7) of *parametric* λ, and the proposed GFGP which applies (11) to the pooling (7). The deep CNNs are trained on ImageNet dataset in the same way as in Sec. 5.1; the detailed procedures to embed the proposed pooling into CNNs and train them are shown in the supplementary material.

The performance results are shown in Table 6a. The *skip* pooling which depends on the local *position* is inferior to the others due to the position sensitivity discussed in

Sec. 5.1.1. And, the stochastic approaches [45, 46] are less effective and especially the S3-pooling [46] which renders further randomness to the pooling is unsuitable for this task on large-scale dataset. Such a randomness in pooling would hamper training of the networks, and the deterministic way works on the large-scale dataset which contains large variation in training images with enough data augmentation. The proposed GFGP favorably outperforms the standard pooling methods as well as the sophisticated ones [30, 21].

We then evaluate the pooling methods on the deeper CNNs of ResNet-50 [12] and ResNeXt-50 [41]. These models contain one *skip* pooling of $(2, 2)$-striding at the first convolution layer, followed by *max*-pooling with 3×3 pool size and $(2, 2)$-striding, and three *skip* pooling of $(2, 2)$-striding in the three ResBlocks, respectively; there are totally five local pooling layers to be replaced with the other pooling methods. In these deeper CNNs, the (original) skip-pooling is again inferior even to the simple avg/max-pooling as in Table 6a. The performance is considerably improved by the proposed GFGP with (7); on ResNet, it is superior even to the further deeper CNN of ResNet-101 which produces 22.48% (top-1) and 6.43% (top-5).

In addition to the above performance comparison, we also extends the previous trainable pooling methods [30, 21] by applying the global feature guidance (Sec. 4) to estimate the trainable parameters via the mapping (11); the DPP [30] contains two parameters per channel while the Gated pooling [21] has one parameter per channel. Our extension[2] favorably boosts the performance as shown in Table 6, demonstrating the generality of the GFGP approach (Fig. 1) .

The proposed method is also evaluated on the Places-365 dataset [48] for scene classification, the different task from the object recognition in the ImageNet. We apply the same CNN models as in Table 6 by replacing all the local pooling layers with our GFGP as well. The performance results in Table 7 demonstrate the effectiveness of the proposed method on the task of scene classification, which shows the applicability of the proposed pooling to versatile tasks.

As shown in the above experimental results, the proposed method generally boosts the performance of deep CNNs by simply replacing the local pooling layers. Since the proposed method operates only on the pooling layers, it is noteworthy that the method could favorably work with the techniques applied to refine the feature map, as discussed in Sec. 5.1.3.

6. Conclusion

In this paper, we have proposed a flexible pooling method adaptively tuned based on input features. The proposed method is composed of both a parameterized pooling function derived from the probabilistic perspective of maximum entropy principle and an adaptive way to estimate the parameters by means of the global feature statistics. The parameters in the pooling function flexibly controls the pooling type toward such as average and max-pooling. And, the global information is effectively incorporated into the local pooling function through adaptively tuning the pooling type (parameters) based on the input global features. We performed thorough experiments to present an effective form of the proposed pooling method and demonstrate favorable performance on large-scale image classification tasks using ImageNet and Places-365 datasets.

[2] As suggested in the papers [30, 21], we employ a exponential activation function for $\mathtt{s}$ in the GFGP-DPP, while the sigmoid is applied to $\mathtt{s}$ in the GFGP-Gated. The other settings in (11) are the same as our GFGP.

Table 6. Performance comparison on ImageNet dataset [6]. The pooling method marked by * is the original setting in the CNN model; skip* in (b) indicates the original setting without manipulating any pooling layers in the models.

(a) VGG-based models

		VGG-16 [32]		VGG-16-mod [17]	
	Pooling	top-1	top-5	top-1	top-5
simple	skip	29.60	10.16	26.00	8.26
	avg	28.44	9.53	25.50	8.01
	max*	27.94	9.25	25.66	7.97
stochastic	stochastic [45]	28.66	9.67	25.74	8.18
	S3 [46]	35.38	13.76	29.45	10.46
trainable	DPP [30]	28.39	9.45	25.55	8.04
	Gate [21]	28.06	9.38	25.20	8.01
	(7) of parametric $\boldsymbol{\lambda}$	27.92	9.19	25.42	7.94
ours	GFGP with (7)	27.17	8.77	24.63	7.50
	GFGP with DPP [30]	28.03	9.23	25.08	7.81
	GFGP with Gate [21]	27.36	9.00	24.82	7.48

(b) ResNet-based models

		ResNet [12]		ResNeXt [41]	
	Pooling	top-1	top-5	top-1	top-5
simple	skip*	23.53	7.00	22.69	6.65
	avg	22.61	6.52	22.14	6.35
	max	22.99	6.71	22.20	6.24
trainable	DPP [30]	22.52	6.63	21.84	5.98
	Gate [21]	22.27	6.33	21.63	5.99
ours	GFGP with (7)	21.79	5.95	21.35	5.74
	GFGP with DPP [30]	22.66	6.60	21.79	6.02
	GFGP with Gate [21]	22.20	6.26	21.45	5.81

Table 7. Performance comparison on Places-365 dataset [48].

(a) VGG-based models

	VGG-16 [32]		VGG-16-mod [17]	
Pooling	top-1	top-5	top-1	top-5
max*	46.25	15.95	45.44	15.11
GFGP with (7)	45.99	15.46	45.33	14.96

(b) ResNet-based models

	ResNet [12]		ResNeXt [41]	
Pooling	top-1	top-5	top-1	top-5
skip*	44.88	14.62	44.52	14.36
GFGP with (7)	44.07	13.94	44.25	13.94

References

[1] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *ICML*, pages 111–118, 2010.

[2] Joan Bruna, Arthur Szlam, and Yann LeCun. Signal recovery from pooling representations. In *ICML*, pages 307–315, 2014.

[3] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *ECCV Workshop on Statistical Learning in Computer Vision*, 2004.

[4] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, pages 764–773, 2017.

[5] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[7] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *ICML*, pages 1319–1327, 2013.

[8] Stephen Grossberg. The quantized geometry of visual space: The coherent computation of depth, form, and lightness. *Behavioral and Brain Sciences*, 6(4):625–657, 1983.

[9] Caglar Gulcehre, Kyunghyun Cho, Razvan Pascanu, and Yoshua Bengio. Learned-norm pooling for deep feedforward and recurrent neural networks. In *ECML PKDD*, pages 530–546, 2014.

[10] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *ICML*, 2017.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[13] Geoffrey E. Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS 2014 Deep Learning Workshop*, 2014.

[14] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, pages 7132–7141, 2018.

[15] David H. Hubel and Torsten Nils Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160:106–154, 1962.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Journal of Machine Learning Research*, 37:448–456, 2015.

[17] Takumi Kobayashi. Analyzing filters toward efficient convnets. In *CVPR*, pages 5619–5628, 2018.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[19] Mandy Lange, Dietlind Zühlke, Olaf Holz, and Thomas Villmann. Applications of lp-norms and their smooth approximations for gradient based learning vector quantization. In *ESANN*, pages 271–276, 2014.

[20] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient- based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] Chen-Yu Lee, Patrick W. Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *AISTATS*, pages 464–472, 2016.

[22] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *ICLR*, 2014.

[23] David G. Lowe. Distinctive image features from scale invariant features. *International Journal of Computer Vision*, 60:91–110, 2004.

[24] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013.

[25] David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002.

[26] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

[27] Steven J. Nowlan and Terrence J. Sejnowski. A selection model for motion processing in area mt of primates. *Journal of Neuroscience*, 15(2):1195–1214, 1995.

[28] Maximilian Riesenhuber and Tomaso Poggio. Just one view: Invariances in inferotemporal cell tuning. In *NIPS*, pages 215–221, 1998.

[29] Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, 1999.

[30] Faraz Saeedan, Nicolas Weber, Michael Goesele, and Stefan Roth. Detail-preserving pooling in deep networks. In *CVPR*, pages 9108–9116, 2018.

[31] Thomas Serre and Tomaso Poggio. A neuromorphic approach to computer vision. *Communications of the ACM*, 53(10):54–61, 2010.

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[33] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[34] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, and Jonathon Shlens. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016.

[35] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *ICCV*, 1998.

[36] Andrea Vedaldi and Karel Lenc. MatConvNet – convolutional neural networks for matlab. In *ACM MM*, 2015.

[37] Feng Wang, Xiang Xiang, Jian Cheng, and Alan L. Yuille. Normface: l_2 hypersphere embedding for face verification. In *ACM MM*, 2017.

[38] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *CVPR*, pages 3360–3367, 2010.

[39] Nicolas Weber, Michael Waechter, Sandra C. Amend, Stefan Guthe, and Michael Goesele. Rapid, detail-preserving image downscaling. *ACM Transaction on Graphic (Proc. SIGGRAPH Asia)*, 35(6):205:1–205:6.

[40] Travis Williams and Robert Li. Wavelet pooling for convolutional neural networks. In *ICLR*, 2018.

[41] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 5987–5995, 2017.

[42] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, pages 1794–1801, 2009.

[43] Dingjun Yu, Hanli Wang, Peiqiu Chen, and Zhihua Wei. Mixed pooling for convolutional neural networks. In *RSKT*, pages 364–375, 2014.

[44] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[45] Matthew D. Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *ICLR*, 2013.

[46] Shuangfei Zhai, Hui Wu, Abhishek Kumar, Yu Cheng, Yongxi Lu, Zhongfei Zhang, and Rogerio Feris. S3pool: Pooling with stochastic spatial sampling. In *CVPR*, pages 770–778, 2017.

[47] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, pages 6848–6856, 2018.

[48] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1452–1464, 2018.